"Signal Generator with Graphical User Interface Control using Wireless Sensor Technology"

Keshavamurthy[1], Dr.Dharmishtan K Varughese[2], Dr. N.J.R Muniraj[3]

[1]Faculty, Atria Institute of Technology, Bangalore, INDIA-560024,
Email: keshavamurthy_s@yahoo.com,Ph: +91- 9844662014
[2] Professors, Department of EC, Karpagam College of Engineering, Coimbatore, INDIA.
[3] Principal, Tejaa Shakthi Institute of Technology for Women, Coimbatore, INDIA.

Abstract — In the fields of communications, signal processing, and in Electrical Engineering more generally, a signal is any time-varying or spatial-varying quantity. In the physical world, any quantity measurable through time or over space can be taken as a signal. Within a complex society, any set of human information or machine data can also be taken as a signal. Signal processing is an area of Electrical Engineering and applied Mathematics that deals with operations on or analysis of signals, in either discrete or continuous time to perform useful operations on those signals. Signals of interest can include sound, images, time-varying measurement values and sensor data, for example biological data such as Electrocardiograms, control system signals, telecommunication transmission signals such as radio signals, and many others. Signals are analog or digital electrical representations of time-varying or spatial-varying physical quantities. So we can also realize how important is signal processing .It can not only convert data from one form (analog) to another (digital) and facilitate its communication and other needs but as well vary parameters provided to practical circuits and appliances So with the context to signals we do face certain problems that we aim to solve: (1). We do find ways to share data with other related hardware and at a distance with modern transmission protocols such as I2C and SPI. (2). Conversion of data from analog to digital and vice versa. (3). Varying power or voltage signal applied to an appliance so that we can save power consumption. (4). The development of a unit that can intelligently read the PC's request at one end and vary the data and signals as and when required.

Key word — Dock light, Zigbee, Nanowatt, I2C Mode, Encrypting, Watchdog and *Graphical User Interface.*

1. Signals generated

In normal embedded development we use input/output signals through I/O Ports, AD Converters, PWM signals and communication protocols I2C, RS232 and SPI. Lot of time is spent on generating above signals / debugging SW modules which deal with the above signals. The embedded signal generator with GUI generates signals from a microcontroller.

2. Block diagram and its description

This family of devices offers the advantages of all PIC18 microcontrollers – namely, high computational performance at an economical price with the addition of high-endurance, Enhanced Flash program memory. In addition to these features, the PIC18F2455/2550/4455/4550 family introduces design enhancements that make these microcontrollers a logical choice for many high-performances, power sensitive applications.

This family of microcontrollers is built with the Nanowatt Technology and hence all of the devices in the PIC18F2455/2550/4455/4550 family incorporate a range of features that can significantly reduce power consumption during operation.

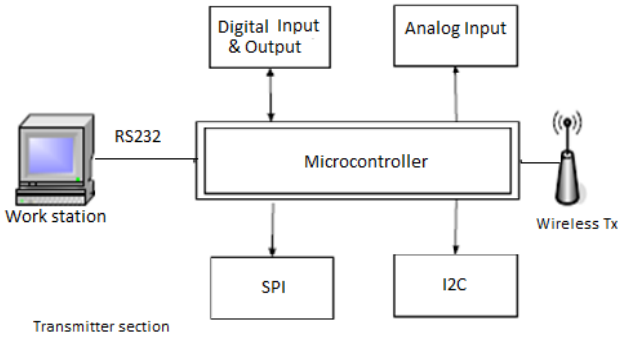

Figure1.1. Transmitter block diagram

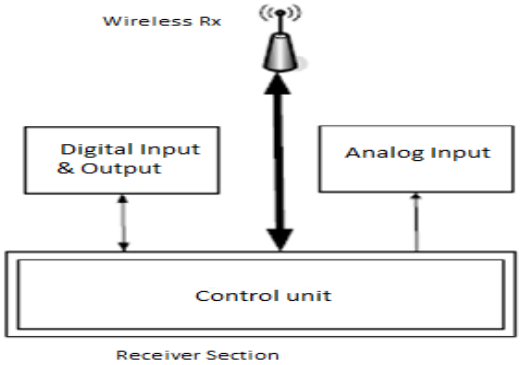

Figure1.2. Receiver block diagram

• **Alternate Run Modes:** By clocking the controller from the Timer1 source or the internal oscillator block, power consumption during code execution can be reduced by as much as 90%. **Multiple Idle Modes:** The controller can also run with its CPU core disabled but the peripherals still active. In these states, power consumption can be reduced even further, to as little as 4%, of normal operation requirements.
• **On-the-Fly Mode Switching:** The power-managed modes are invoked by user code during operation, allowing the user to incorporate power-saving ideas into their applications software design.
• **Low Consumption in Key Modules:** The power requirements for both Timer1 and the Watchdog Timer are minimized.
Devices in the PIC18F2455/2550/4455/4550 family incorporate a fully featured Universal Serial Bus Communications module that is compliant with the USB Specification Revision 2.0. The module supports both low-speed and full-speed communication for all supported data transfer types. It also incorporates its own on-chip transceiver and 3.3V regulator and supports the use of external transceivers and voltage regulators. All of the devices in the PIC18F2455/2550/4455/4550 family offer twelve different oscillator options, allowing users a wide range of choices in developing application hardware. Besides its availability as a clock source, the internal oscillator block provides a stable reference source that gives the family additional features for robust operation:

• **Fail-Safe Clock Monitor:** This option constantly monitors the main clock source against a reference signal provided by the internal oscillator. If a clock failure occurs, the controller is switched to the internal oscillator block, allowing for continued low-speed operation or a safe application shutdown.
• **Two-Speed Start-up:** This option allows the internal oscillator to serve as the clock source from Power-on Reset, or wake-up from Sleep mode, until the primary clock source is available.

Port A: PORT-A is an 8-bit wide, bidirectional port. The corresponding data direction register is TRISA. Setting a TRISA bit (= 1) will make the corresponding PORTA pin an input (i.e., put the corresponding output driver in a high-impedance mode). Clearing a TRISA bit (= 0) will make the corresponding PORTA pin an output (i.e., put the contents of the output latch on the selected pin).

Reading the PORTA register reads the status of the pins, whereas writing to it, will write to the port latch. The Data Latch (LATA) register is also memory mapped. Read-modify-write operations on the LATA register read and write the latched output value for PORTA.

The other PORTA pins are multiplexed with analog inputs, the analog VREF+ and VREF- inputs and the comparator voltage reference output. The operation of pins RA3:RA0 and RA5 as A/D converter inputs is selected by clearing or setting the control bits in the ADCON1 register (A/D Control Register 1). Pins RA0 through RA5 may also be used as comparator inputs or outputs by setting the appropriate bits in the CMCON register.

Port B: PORTB is an 8-bit wide, bidirectional port. The corresponding data direction register is TRISB. Setting a TRISB bit (= 1) will make the corresponding PORTB pin an input (i.e., put the corresponding output driver in a high-impedance mode). Clearing a TRISB bit (= 0) will make the corresponding PORTB pin an output (i.e., put the contents of the output latch on the selected pin). The Data Latch register (LATB) is also memory mapped. Read-modify-write operations on the LATB register read and write the latched output value for PORTB.

Each of the PORTB pins has a weak internal pull-up. A single control bit can turn on all the pull-ups. This is performed by clearing bit, RBPU (INTCON2<7>). The weak pull-up is automatically turned off when the port pin is configured as an output. The pull-ups are disabled on a Power-on Reset.
Port C: PORTC is an 8-bit wide, bidirectional port. The corresponding data direction register is TRISC. Setting a TRISC bit (= 1) will make the corresponding PORTC pin an input (i.e., put the corresponding output driver in a high-impedance mode). Clearing a TRISC bit (= 0) will make the corresponding PORTC pin an output (i.e., put the contents of the output latch on the selected pin).

The Data Latch register (LATC) is also memory mapped. Read-modify-write operations on the LATC register read and write the latched output value for PORTC. PORTC is multiplexed with several peripheral functions. The pins have Schmitt Trigger input buffers. RC1 is normally configured by configuration bit, CCP2MX, as the default peripheral pin of the CCP2 module.
Port D: PORTD is an 8-bit wide, bidirectional port. The corresponding data direction register is TRISD. Setting a TRISD bit (= 1) will make the corresponding PORTD pin an input (i.e., put the corresponding output driver in a high-impedance mode). Clearing a TRISD bit (= 0) will make the corresponding PORTD pin an output (i.e., put the contents of the output latch on the selected pin). The Data Latch register (LATD) is also memory mapped.

Read-modify-write operations on the LATD register read and write the latched output value for PORTD. All pins on PORTD are implemented with Schmitt Trigger input buffers.

Each pin is individually configurable as an input or output. Three of the PORTD pins are multiplexed with outputs P1B, P1C and P1D of the Enhanced CCP module.

Port E: For 40/44-pin devices, PORTE is a 4-bit wide port. Three pins (RE0/RD/AN5, RE1/WR/AN6 and RE2/CS/ AN7) are individually configurable as inputs or outputs. These pins have Schmitt Trigger input buffers. When selected as an analog input, these pins will read as '0's. The corresponding data direction register is TRISE. Setting a TRISE bit (= 1) will make the corresponding PORTE pin an input (i.e., put the corresponding output driver in a high-impedance mode). Clearing a TRISE bit (= 0) will make the corresponding PORTE pin an output (i.e., put the contents of the output latch on the selected pin). TRISE controls the direction of the RE pins, even when they are being used as analog inputs. The user must make sure to keep the pins configured as inputs when using them as analog inputs. The upper four bits of the TRISE register also control the operation of the Parallel Slave Port.

3.Master synchronous serial port (MSSP) module

The Master Synchronous Serial Port (MSSP) module is a serial interface, useful for communicating with other peripheral or microcontroller devices. These peripheral devices may be serial EEPROMs, shift registers,display drivers, A/D converters, etc.

The MSSP module has three associated control registers.These include a status register (SSPSTAT) and two control registers (SSPCON1 and SSPCON2). The use of these registers and their individual Configuration bits differ significantly depending on whether the MSSP module is operated in SPI or I2C mode.

3.1. Registers and its initializations

The MSSP module has four registers for SPI mode operation. These are:

- MSSP Control Register 1 (SSPCON1)
- MSSP Status Register (SSPSTAT)
- Serial Receive/Transmit Buffer Register (SSPBUF)
- MSSP Shift Register (SSPSR) – Not directly accessible

SSPCON1 and SSPSTAT are the control and status registers in SPI mode operation. The SSPCON1 register is readable and writable. The lower six bits of the SSPSTAT are read-only.

The upper two bits of the SSPSTAT are read/ write. SSPSR is the shift register used for shifting data in or out. SSPBUF is the buffer register to which data bytes are written to or read from .In receive operations, SSPSR and SSPBUF together create a double-buffered receiver. When SSPSR receives a complete byte, it is transferred to SSPBUF and the SSPIF interrupt is set .During transmission, the SSPBUF is not double buffered .A write to SSPBUF will write to both SSPBUF and SSPS

3.2. SPI Master/slave Connection Typical Connection

The master controller (Processor 1) initiates the data transfer by sending the SCK signal .Data is shifted out of both shift registers on their programmed clock edge and latched on the opposite edge of the clock.

Both processors should be programmed to the same Clock Polarity (CKP), then both controllers would send and receive data at the same time. Whether the data is meaningful (or dummy data) depends on the application software. This leads to three scenarios for data transmission:

- Master sends data – Slave sends dummy data
- Master sends data – Slave sends data.
- Master sends dummy data – Slave sends data.

3.2.1. Master and slave mode

In Master mode, the data is transmitted /received as soon as the SSPBUF register is written to. If the SPI is only going to receive, the SDO output could be disabled (programmed as an input). The SSPSR register will continue to shift in the signal present on the SDI pin at the programmed clock rate. As each byte is received , it will be loaded into the SSPBUF register as if a normal received byte (interrupts and status bits appropriately set). This could be useful in receiver applications as a "Line Activity Monitor" mode.

In Slave mode, the data is transmitted and received as the external clock pulses appear on SCK. When the last bit is latched, the SSPIF interrupt flag bit is set. While in Slave mode, the external clock is supplied by the external clock source on the SCK pin.

This external clock must meet the minimum high and low times as specified in the electrical specifications .While in sleep mode, the slave can transmit/receive data. When a byte is received, the device can be configured to wake- up from Sleep.

3.2.2. SPI Master/slave Connection

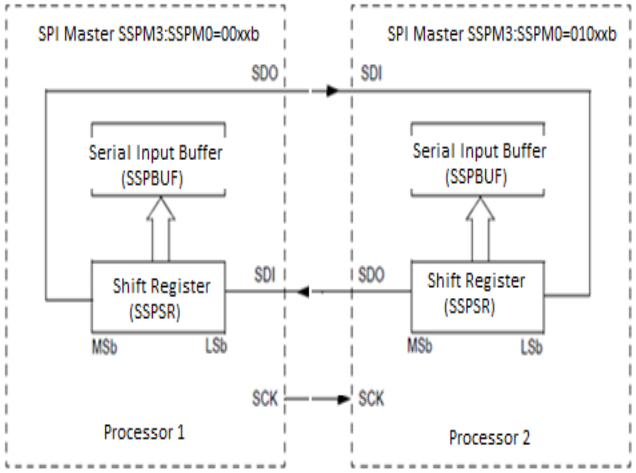

Figure3.1. SPI connection

3.2.3. I2C Mode

The MSSP module in I2C mode fully implements all master and slave functions (including general call support) and provides interrupts on Start and Stop bits in hardware to determine a free bus (multi-master function). The MSSP module implements the standard mode specifications, as well as 7-bit and 10-bit addressing .Two pins are used for data transfer:

- Serial clock (SCL) – RB1/AN10/INT1/SCK/SCL
- Serial data (SDA) – RB0/AN12/INT0/FLT0/SDI/SDA

The user must configure these pins as inputs by setting the associated TRIS bits.

3.2.4. Capture/Compare/PWM (CCP) module

PIC18F2455/2550/4455/4550 devices all have two CCP (Capture/Compare/PWM) modules. Each module contains a 16-bit register, which can operate as a 16-bit Capture register, a 16-bit Compare register or a PWM Master/Slave Duty Cycle register.

3.2.5. CCP Module Configuration

Each Capture/Compare/PWM module is associated with a control register (generically, CCP x CON) and a data register (CCP Rx). The data register, in turn, is comprised of two 8-bit registers: CCP Rx L (low byte) and CCP Rx H (high byte). All registers are both readable and writable.

The CCP modules utilize Timers 1, 2 or 3, depending on the mode selected. Timer1 and Timer3 are available to modules in Capture or Compare modes, while Timer2 is available for modules in PWM mode.

CCP MODE – TIMER RESOURCE

CCP/ECCP Mode	Timer Resource
Capture	Timer1 or Timer3
Compare	Timer1 or Timer3
PWM	Timer2

3.2.6. Capture Mode

In Capture mode, the CCPR x H : CCPR x L register pair captures the 16-bit value of the TMR1 or TMR3 registers when an event occurs on the corresponding CCP x pin.

The event is selected by the mode select bits,CCPxM3:CCPxM0 (CCP x CON<3:0>). When a capture is made, the interrupt request flag bit, CCP x IF, is set; it must be cleared in software. If another capture occurs before the value in register CCPRx is read, the old captured value is overwritten by the new captured value.

3.3. PWM Generation

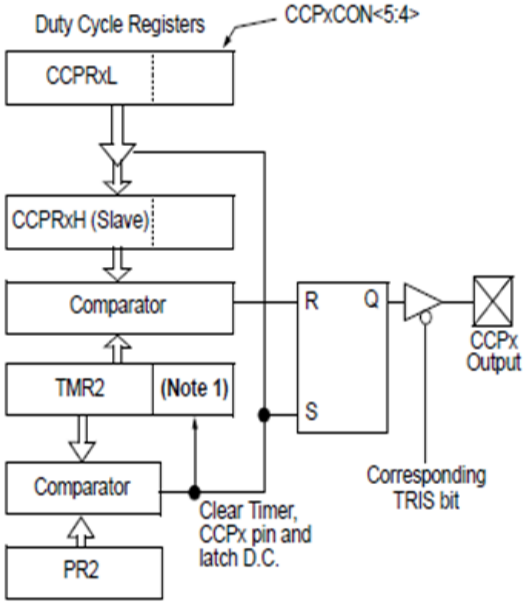

Figure3.2. PWM generator block diagram.

3.3.1. PWM Output

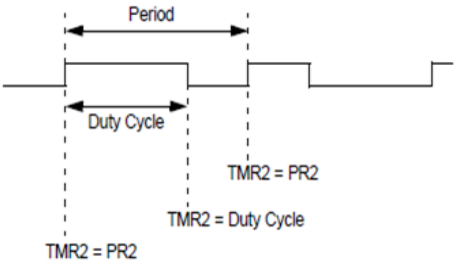

Figure3.2. PWM generation blockdiagram and PWM output.

In Pulse-Width Modulation (PWM) mode:

The CCPx pin produces up to a 10-bit resolution PWM output. A PWM output has a time base (period) and a time that the output stays high (duty cycle).The frequency of the PWM is the inverse of the period (1/period).

3.3.1. PWM period and duty cycle

The PWM period is specified by writing to the PR2 register which is given by

$$\text{PWM Period} = [(PR2) + 1] \cdot 4 \cdot TOSC \cdot (\text{TMR2 Prescale Value})$$

PWM frequency is defined as 1/[PWM period].

When TMR2 is equal to PR2, the following three events occur on the next increment cycle:

TMR2 is cleared

• The CCPx pin is set (exception: if PWM duty cycle = 0%, the CCPx pin will not be set)

• The PWM duty cycle is latched from CCPRxL into CCPRxH.

- The PWM duty cycle is specified by writing to the CCPRxL register and to the CCPxCON<5:4> bits.

Up to 10-bit resolution is available.

- The CCPRxL contains the eight MSBs and the CCPxCON<5:4> bits contain the two LSBs.
- CCPRxL and CCPxCON<5:4> can be written to at anytime.But the duty cycle value is not latched intoCCPRxH until after a match between PR2 and TMR2 occurs (i.e., the period is complete).
- In PWM mode, CCPRxH is a read-only register.
- If the PWM duty cycle value is longer than the PWM period, the CCPx pin will not be cleared.

The CCPRxH register and a 2-bit internal latch are used to double-buffer the PWM duty cycle. This double-buffering is essential for glitch less PWM operation.

When the CCPRxH and 2-bit latch match TMR2, concatenated with an internal 2-bit Q clock or 2 bits of the TMR2 presale, the CCPx pin is cleared. The maximum PWM resolution (bits) for a given PWM frequency is given by the equation:

$$\text{PWM Resolution (max)} = \frac{\log\left(\frac{FOSC}{FPWM}\right)}{\log(2)} \text{bits}$$

4. Zigbee RF module

The XBee/XBee-PRO ZB RF Modules are designed to operate within the ZigBee protocol and support the unique needs of low-cost, low-power wireless sensor networks. The modules require minimal power and provide reliable delivery of data between remote devices. The modules operate within the ISM 2.4 GHz frequency band and are compatible with the following: •XBee RS-232 Adapter •XBee RS-485 Adapter •XBee Analog I/O Adapter •XBee Digital I/O Adapter •XBee Sensor •XBee USB Adapter •XStick •Connect Port X Gateways •XBee Wall Router.

Metal objects next to the antenna or between transmitting and receiving antennas can often block or reduce the transmission distance. Some objects that are often overlooked are metal poles, metal studs or beams in structures, concrete (it is usually reinforced with metal rods), metal enclosures, vehicles, elevators, ventilation ducts.

4.1. Serial Communications

The XBee RF Modules interface to a host device through a logic-level asynchronous serial port. Through its serial port, the module can communicate with any logic and voltage compatible UART; or through a level translator to any serial device (for example: through a RS-232 or USB interface board). The XBee modules maintain small buffers to collect received serial and RF data. The serial receive buffer collects incoming serial characters and holds them until they can be processed. The serial transmit buffer collects data that is received via the RF link that will be transmitted out the UART

4.2 Serial Receive Buffer

When serial data enters the RF module through the DIN Pin (pin 3), the data is stored in the serial receive buffer until it can be processed. Under certain conditions, the module may not be able to process data in the serial receive buffer immediately. If large amounts of serial data are sent to the module, CTS flow control may be required to avoid overflowing the serial receive buffer.

Cases in which the serial receive buffer may become full and possibly overflow:

1. If the module is receiving a continuous stream of RF data, the data in the serial receive buffer will not be transmitted until the module is no longer receiving RF data.

2. If the module is transmitting an RF data packet, the module may need to discover the destination address or establish a route to the destination. After transmitting the data, the module may need to retransmit the data if an acknowledgment is not received, or if the transmission is a broad-cast. These issues could delay the processing of data in the serial receive buffer.

4.3. Serial Transmit Buffer

When RF data is received, the data is moved into the serial transmit buffer and sent out the UART. If the serial

transmit buffer becomes full enough such that all data in a received RF packet won't fit in the serial transmit buffer, the entire RF data packet is dropped.

Cases in which the serial transmit buffer may become full resulting in dropped RF packets

1. If the RF data rate is set higher than the interface data rate of the module, the module could receive data faster than it can send the data to the host.
2. If the host does not allow the module to transmit data out from the serial transmit buffer because of being held off by hardware flow control.

CTS Flow Control and RTS Flow Control

If CTS flow control is enabled (D7 command), when the serial receive buffer is 17 bytes away from being full, the module de-asserts CTS (sets it high) to signal to the host device to stop sending serial data. CTS is re-asserted after the serial receive buffer has 34 bytes of space.

4.4. Modes of Operation

4.4.1 Idle Mode

When not receiving or transmitting data, the RF module is in Idle Mode. The module shifts into the other modes of operation under the following conditions: •Transmit Mode (Serial data in the serial receive buffer is ready to be packetized) •Receive Mode (Valid RF data is received through the antenna) •Sleep Mode (End Devices only) •Command Mode (Command Mode Sequence is issued

4.4.2. Transmit Mode

When serial data is received and is ready for packetization, the RF module will exit Idle Mode and attempt to transmit the data. The destination address determines which node(s) will receive the data. Prior to transmitting the data, the module ensures that a 16-bit network address and route to the destination node have been established. If the destination 16-bit network address is not known, network address discovery will take place.

If a route is not known, route discovery will take place for the purpose of establishing a route to the destination node. If a module with a matching network address is not discovered, the packet is discarded. The data will be transmitted once a route is established. If route discovery fails to establish a route, the packet will be discarded.

4.5. ZigBee Security Model

ZigBee security is applied to the Network and APS layers. Packets are encrypted with 128-bit AES encryption. A network key and optional link key can be used to encrypt data. Only devices with the same keys are able to communicate together in a network. Routers and end devices that will communicate on a secure network must obtain the correct security keys.

ZigBee end devices are intended to be battery-powered devices capable of sleeping for extended periods of time. Since end devices may not be awake to receive RF data at a given time, routers and coordinators are equipped with additional capabilities (including packet buffering and extended transmission timeouts) to ensure reliable data delivery to end devices. ZigBee defines a trust center device that is responsible for authenticating devices that join the network. The trust center also manages link key distribution in the network.

4.5.1. Message integrity Code

If APS security is enabled, the APS header and data payload are authenticated with 128-bit AES. A hash is performed on these fields and appended as a 4-byte message integrity code (MIC) to the end of the packet. This MIC is different than the MIC appended by the network layer. The MIC allows the destination device to ensure the message has not been changed. If the destination device receives a packet and the MIC does not match the destination device's own hash of the data, the packet is dropped.

4.5.2. Network Layer Security

The network key is used to encrypt the APS layer and application data. In addition to encrypting application messages, network security is also applied to route request and reply messages, APS commands, and ZDO commands. Network encryption is not applied to MAC layer transmissions such as beacon transmissions, etc. If security is enabled in a network, all data packets will be encrypted with the network key. Packets are encrypted and authenticated using 128-bit AES. This is shown in the figure below.

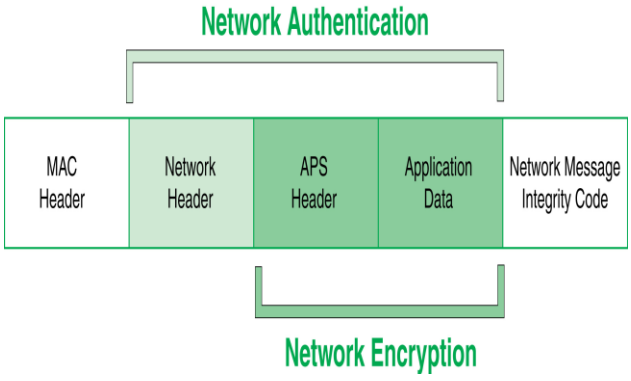

Figure 4: Network security layer

5. Software used

5.1. C# Programming Language

C# is an object-oriented programming language developed by Microsoft Corporation. C# source code as well as those of other .NET languages is compiled into an intermediate byte code called Microsoft Intermediate Language. C# is primarily derived from the C, C++, and Java programming languages with some features of Microsoft's Visual Basic in the mix. C# is used to develop applications for the Microsoft .NET environment. .NET offers an alternative to Java development. Microsoft's Visual Studio .NET development environment incorporates several different languages including ASP.NET, C#, C++, and J# (Microsoft Java for .NET), all of which compile to the Common Language Runtime.

The C# code that we write is supported on Microsoft Visual studio 2008 and we handle all our operations by this code. Through this code we not only open the serial port and perform input and output operations but also receive analog values and cause their conversion to digital values and display them as well as transmit them to the microcontroller.

What is most important is that through this code we are able to vary the frequency and duty cycle of our PWM signal and also send the microcontroller the changes so that it can communicate with the remote terminal where the power of the appliance can be varied.

5.2. IDE: MPLAB from microchip

MPLAB X IDE is a software program that runs on a PC (Windows®, Mac OS®, Linux®) to develop applications for Microchip microcontrollers and digital signal controllers. It is called an Integrated Development Environment, or IDE, because it provides a single integrated "environment" to develop code for embedded microcontrollers.

The peripherals and the amount of memory an application needs to run a program largely determines which PICmicro MCU to use. Other factors might include the power consumed by the microcontroller and its "form factor," i.e., the size and characteristics of the physical package that must reside on the target design.

A development system for embedded controllers is a system of programs running on a desktop PC to help write, edit, debug and program code – the intelligence of embedded systems applications – into a microcontroller. MPLAB IDE, runs on a PC and contains all the components needed to design and deploy embedded systems applications.

5.3. Used in Testing

Once the code builds with no errors, it needs to be tested. MPLAB IDE has components called "debuggers" and free software simulators for all PICmicro and dsPIC devices to help test the code. Even if the hardware is not yet finished, you can begin testing the code with the simulator, a software program that simulates the execution of the microcontroller. The simulator can accept a simulated input (stimulus), in order to model how the firmware responds to external signals. The simulator can measure code execution time, single-step through code to watch variables and peripherals, and trace the code to generate a detailed record of how the program ran.

Once the hardware is in a prototype stage, a hardware debugger, such as MPLAB ICE or MPLAB ICD 2 can be used. These debuggers run the code in real time on your actual application. The MPLAB ICE physically replaces the microcontroller in the target using a high-speed probe to give you full control over the hardware in your design.

The MPLAB ICD 2 uses special circuitry built into many Microchip MCUs with Flash program memory and can "see into" the target microcontrollers program and data memory. The MPLAB ICD 2 can stop and start program execution, allowing you to test the code with the microcontroller in place on the application.

5.4. Microsoft Visual Studio 2008

Microsoft Visual Studio is an integrated development environment (IDE) from Microsoft. It can be used to develop console and graphical user interface applications along with Windows Forms applications, web sites, web applications, and web services in both native code together with managed code for all platforms supported by Microsoft Windows, Windows Mobile, Windows CE, .NET Framework, .NET Compact Framework and Microsoft Silver light.

Visual Studio includes a code editor supporting IntelliSense as well as code refactoring. The integrated debugger works both as a source-level debugger and a machine-level debugger.

It accepts plug-ins that enhance the functionality at almost every level—including adding support for source-control systems (like Subversion and Visual SourceSafe) and adding new toolsets like editors and visual designers for domain-specific languages or toolsets for other aspects of the software development lifecycle.

Features

5.5 Code editor

Visual Studio, like any other IDE, includes a code editor that supports syntax high lighting and code completion using IntelliSense for not only variables, functions and methods but also language constructs like loops and queries.

IntelliSense is supported for the included languages, as well as for XML and for Cascading Style Sheets and JavaScript when developing web sites and web applications. Auto complete suggestions are popped up in a modeless list box, overlaid on top of the code editor. In Visual Studio 2008 onwards, it can be made temporarily semi-transparent to see the code obstructed by it. The code editor is used for all supported languages.

5.6. Debugger

Visual Studio includes a debugger that works both as a source-level debugger and as a machine-level debugger. It works with both managed code as well as native code and can be used for debugging applications written in any language supported by Visual Studio. In addition, it can also attach to running processes and monitor and debug those processes. If source code for the running process is available, it displays the code as it is being run. If source code is not available, it can show the disassembly. The Visual Studio debugger can also create memory dumps as well as load them later for debugging. Multi-threaded programs are also supported. The debugger can be configured to be launched when an application running outside the Visual Studio environment crashes.

5.7. Docklight RS232 Terminal

Docklight can send out user-defined sequences according to the protocol used and it can react to incoming sequences. This makes it possible to simulate the behavior of a serial communication device, which is particularly useful for generating test conditions that are hard to reproduce with the original device (e.g. problem conditions).

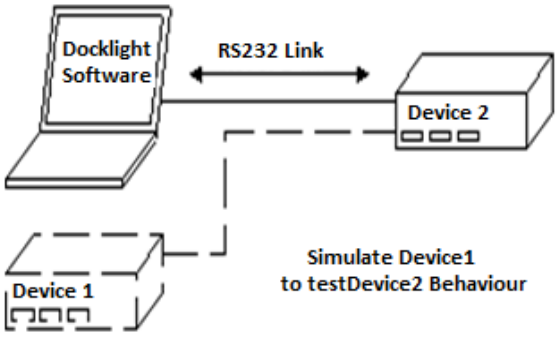

Figure 5.1.Simulated device

5.7. Detecting specific data sequences

In many test cases you will need to check for a specific sequence within the RS232 data that indicates a problem condition. Dock light manages a list of such data sequences for you and is able to perform user-defined actions after detecting a sequence, e.g. taking a snapshot of all communication data before and after the error message was received.

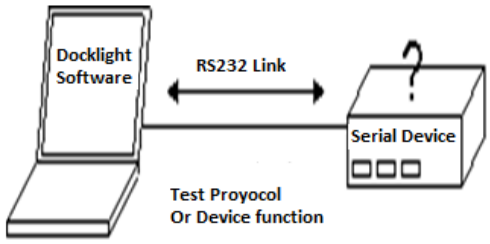

Figure5.2. Test protocol

5.8. Logging RS232 data

All serial communication data can be logged using two different file formats: use plain text format for fast logging and storing huge amounts of data. Or create a HTML file with styled text that lets you easily distinguish between incoming and outgoing data or additional information.

6. Future enhancement

1. With advancements in Zigbee protocol, the range which is a major operational barrier can be increased.
2. Along with the Zigbee Coordinator and Zigbee End Device, if we connect a router then our setup can work for many remote terminals.
3. They can be used in industrial control rooms in order to handle the appliances by sending PWM signals from a distance and use this idea in a productive manner.
4. It is really effective as a control device when the end applications are to be controlled from a distance i.e. in case of harsh environments. Ex: industrial heaters, lighting etc.
5. Wireless measuring of analog data for example as we can connect a temperature sensor cum logger, gas sensor etc.
6. We can achieve a better resolution by increasing it up to 10 bit.
7. Since it has a better range as we use Zigbee, our design can perform switching application to appliances from a distance. Data can be read from and written to as we need for Digital Processing.

9.4 CONCLUSION

The signal generated out of this module can be used in any 8/16/32 bit embedded system for rapid prototyping/ debugging. This unit has the capability to vary the parameters of the PWM by which we can vary the signal power supplied to the end users. More flexibility in our approach as we can modify the execution code as and when needed to add new utilities. Less hardware and time involved in the generation of PWM than

conventional ways. Apart from this the unit will also have the capability to monitor the digital I/O lines and analog values from the remote terminal.

Today we have learned to upgrade our daily life applications with new technologies. We have made an effort to provide a design which can be used in a number of ways as per the requirement. It can provide industrial applications as well as being equally effective in a simple house hold. For example, we can use our simple unit to control a heater at home. It can set two thresholds and by the help of our temperature sensor, we can detect if the temperature has gone up to the upper threshold and by the help of the relays in our design we can switch off the heater. So it comes so handy in a simple household.

Similarly, we have control application of our unit in industrial heating and gas furnaces, lighting, motors etc. From a distant control room, we can vary the power of the appliances as well as monitor the temperature and gas levels etc and according to our real time deadlines, can turn it off using the relays or reduce power applied to it by using our PWM module.

Hence we see how our embedded unit handles a lot of functionalities with simplified hardware and with a high accuracy rate. It also reduces manpower and hazardous results. So we have achieved what we had aimed for and we can really put it to use effectively.

BIBLIOGRAPHY

[1] T. Kikuchi, T. Kenjo, and S. Fukuda, "Remote laboratory for a brushless dc motor," IEEE Trans. Educ., vol. 44, May 2001.

[2] A. Hattori, N. Suzuki, S. Suzuki, A. Takatsu, M. P. Bauer, A. Hirner, S.Takahashi, S. Kobayashi, Y. Yamazaki, Y. Adachi, T. Kumano, and A.Ikemoto, "Tele-virtual surgery with sharing tactile sensations between Japan and Germany," in Proc. 22nd Annu. Int. Conf. IEEE Engineering Medicine Biology Soc., Chicago, IL, July 23–28, 2000, pp. 2423–2425.

[3] Palloff R. and Pratt K., The Virtual Student - A Profile and Guide to Working with Online Learners, Jossey - Bass, 2003, ISBN 0-7879- 6474-3

[4] I.A. Grout, J.Walsh, T. O'Shea and M. Canavan, A Local and Remote Laboratory User Experimentation Access Arrangement using Digital Programmable Logic, Proceedings of the Remote Engineering and Virtual Instrumentation Symposium (REV 2004), September 2004, Villach, Austria

[5] J. Murphy, I. Grout, J. Walsh, and T. O'Shea, Local and Remote Laboratory User Experimentation Access using Digital Programmable Logic, International Journal of Online Engineering, vol. 1, October 2005, http://www.i-joe.com

[6] Rohrig, C. and Jochheim, A., The Virtual Lab for controlling real experiments via Internet, Proceedings of the IEEE International Symposium on Computer Aided Control System Design, 1999, pp 279-284

[7] Best R. E., Phase-locked loops: theory, design, and applications, 1984,ISBN 0070050503

[8] Rohde U.L., Digital PLL frequency synthesizers: theory and design, Prentice-Hall, London, 1983, ISBN 0132142392

[9] Burbidge, M.J. and Richardson, A.M., Simple digital test approach for embedded charge-pump phase-locked loops, Electronics Letters, Volume 37, Issue 22, 25th October 2001, pp 1318-1319

[10]. ieeexplore.ieee.org (for information on signal generators)

[11]http://www.matrixmultimedia.com,www.sparkfun.com/datasheets/Wireless/Zigbee/XBee-2.5-Datasheet (for Zigbee module)

[12]www.sourcecodeonline.com/list?q=embedded_system_projects_on_ieee_paper

[13].www.ieeepaper.com/free-ieee-papers-embedded-system-31.htm

[14]http://www.alibaba.com/RS232